4/6/2021
Ken Amamori
CS 4375

Deadlock Report

Instruction:

Take some code that you have already written, for this class or some other class, and modify it so that it can deadlock.
Submit a brief report with your code and evidence that it actually does deadlock.

Explanation:

I used the previous assignment's code (concurrency beyond locks) in order to complete this assignment. The modification I made is on condition for putChar while loop. I changed it so that it will go into while-loop, which will cause to wait() while the other is waiting as well. The code satisfies the Conditions for Deadlock, which are listed below with an explanation.

Condition 1: Circular Wait

In the code, two conditional variables work as a lock. One we do wait() in the while-loop, and the other will do the same. As a result, both of them will have the ability to release the lock; however, both of them are in wait(). Thus, it is a circular wait(). One is waiting for the other.

Condition 2: Hold and Wait

In the code, there are two conditional variables, and one variable (lock with conditional variable) is holding the resources required for the other process to proceed. For example, charAdded holds the recourses that enable to release charRemoved's processes.

Condition 3: No Preemption

There is no priority in executing any of the code in the code and cannot forcibly remove or add any type of processes threads that are holding them to wait.

Condition 4: Mutual Exclusion

In the code, two conditional variables work as a lock. Each variable has control over which part, which method to run since if one is running, it will run a piece of code that the other conditional variable has no control. Thus, each lock (conditional variable) has exclusive control of resources that are required.

Result:

In the code, I have added four print statements in order to see where visually what is taking place in the code. I added print statements right after entering into while-loop in putChar() and getChar(). Other two print statements are right after the while-loops in putChar() and getChar(). As figure 1 (placed below) shows, only two print statements that I added were displayed. Both of them are that was inside of while-loop, meaning it never broke neither while-loop. It shows that both locks were never unlocked and stayed at the wait, and it will be at wait forever since no one will notify (unlock). On the other hand, as you can see in Figure 2 (placed below), it repeatedly enters to while-loop, goes outside of the while-loop, and retrieves the character. This example is when it is not deadlocked.

4/6/2021
Ken Amamori
CS 4375

Code:

deadlock.py (Modified):

```
1.  #!/usr/bin/env python3
2.  # pump.py  Nigel Ward, 2021.  Based on John Osterhout's Producer/Consumer code.
3.
4.  import sys, threading, time
5.  from threading import Condition
6.
7.  global count, putIndex,  getIndex, cbuffer, bufLock
8.
9.  def pumpProducer():
10.     print('starting Producer')
11.     arpaciQuote ='" Yeats famously said "Education is not the filling of a pail but the lighting of a fire." He was
        right but wrong at the same time. You do have to "fill the pail" a bit, and these notes are certainly here to
        help with that part of your education; after all, when you go to interview at Google, and they ask you a trick
        question about how to use semaphores, it might be good to actually know what a semaphore is, right? But
        Yeats's larger point is obviously on the mark: the real point of education is to get you interested in
        something, to learn something more about the subject matter on your own and not just what you have to
        digest to get a good grade in some class. As one of our fathers (Remzi's dad, Vedat Arpaci) used to say,
        "Learn beyond the classroom".'
12.     for charToSend in arpaciQuote:
13.         putChar(charToSend)
14.     putChar(0)    # the solution!!
15.
16.
17. def putChar(character):
18.     global count, putIndex, bufLock, charRemoved, charAdded
19.     bufLock.acquire()
20.     while count == 0:
21.         print('removed')
22.         charRemoved.wait()
23.     print('out of removed')
24.     count += 1
25.     cbuffer[putIndex] = character
26.     putIndex += 1
```

```
27.     if putIndex == bufsize:
28.         putIndex = 0
29.     charAdded.notify()
30.     bufLock.release()
31.
32. def pumpConsumer():
33.     print('starting Consumer')
34.     while (1):
35.         for i in range(100):
36.             c = getChar()
37.             if c == 0:
38.                 return
39.             print(c,end="")
40.         print("")  # newline
41.
42. def getChar():
43.     global count, getIndex, bufLock, charRemoved, charAdded
44.     bufLock.acquire()
45.     while (count == 0):
46.         print('added')
47.         charAdded.wait()
48.     print('out of added')
49.     count -= 1
50.     c = cbuffer[getIndex]
51.     getIndex += 1
52.     if (getIndex == bufsize):
53.         getIndex = 0
54.     charRemoved.notify()
55.     bufLock.release()
56.     return c
57.
58. ### main ###
59. if len(sys.argv) != 2:
60.     print(len(sys.argv))
61.     print("usage: pump bufferSize")
62.     exit(1)
```

```
63.
64.  bufsize = int(sys.argv[1])
65.  cbuffer = ['x'] * bufsize    # circular buffer; x means uninitialized
66.  count = putIndex = getIndex = 0
67.  bufLock = threading.Lock()
68.  charAdded = Condition(bufLock)
69.  charRemoved = Condition(bufLock)
70.
71.  consumer = threading.Thread(target=pumpConsumer)
72.  consumer.start()
73.
74.  producer = threading.Thread(target=pumpProducer)
75.  producer.start()
76.
77.  producer.join()
78.
```

pump-fixed.py (original):

```
1.   #!/usr/bin/env python3
2.   # pump.py  Nigel Ward, 2021.  Based on John Osterhout's Producer/Consumer code.
3.
4.   import sys, threading, time
5.   from threading import Condition
6.
7.   global count, putIndex,  getIndex, cbuffer, bufLock
8.
9.   def pumpProducer():
10.      print('starting Producer')
11.      arpaciQuote ='" Yeats famously said "Education is not the filling of a pail but the lighting of a fire." He was
     right but wrong at the same time. You do have to "fill the pail" a bit, and these notes are certainly here to
     help with that part of your education; after all, when you go to interview at Google, and they ask you a trick
     question about how to use semaphores, it might be good to actually know what a semaphore is, right? But
     Yeats's larger point is obviously on the mark: the real point of education is to get you interested in
     something, to learn something more about the subject matter on your own and not just what you have to
     digest to get a good grade in some class. As one of our fathers (Remzi's dad, Vedat Arpaci) used to say,
     "Learn beyond the classroom".'
```

```
12.     for charToSend in arpaciQuote:
13.         putChar(charToSend)
14.     putChar(0)    # the solution!!
15.
16.
17.  def putChar(character):
18.     global count, putIndex, bufLock, charRemoved, charAdded
19.     bufLock.acquire()
20.     while count >= bufsize:
21.         charRemoved.wait()
22.     count += 1
23.     cbuffer[putIndex] = character
24.     putIndex += 1
25.     if putIndex == bufsize:
26.         putIndex = 0
27.     charAdded.notify()
28.     bufLock.release()
29.
30.  def pumpConsumer():
31.     print('starting Consumer')
32.     while (1):
33.         for i in range(100):
34.             c = getChar()
35.             if c == 0:
36.                 return
37.             print(c,end="")
38.         print("")  # newline
39.
40.  def getChar():
41.     global count, getIndex, bufLock, charRemoved, charAdded
42.     bufLock.acquire()
43.     while (count == 0):
44.         charAdded.wait()
45.     count -= 1
46.     c = cbuffer[getIndex]
47.     getIndex += 1
```

```
48.     if (getIndex == bufsize):
49.         getIndex = 0
50.     charRemoved.notify()
51.     bufLock.release()
52.     return c
53.
54. ### main ###
55. if len(sys.argv) != 2:
56.     print(len(sys.argv))
57.     print("usage: pump bufferSize")
58.     exit(1)
59.
60. bufsize = int(sys.argv[1])
61. cbuffer = ['x'] * bufsize    # circular buffer; x means uninitialized
62. count = putIndex = getIndex = 0
63. bufLock = threading.Lock()
64. charAdded = Condition(bufLock)
65. charRemoved = Condition(bufLock)
66.
67. consumer = threading.Thread(target=pumpConsumer)
68. consumer.start()
69.
70. producer = threading.Thread(target=pumpProducer)
71. producer.start()
72.
73. producer.join()
74.
```

4/6/2021
Ken Amamori
CS 4375

Output:

```
Kens-MBP:deadlock ken$ python3 deadlock.py 10
starting Consumer
added
starting Producer
removed
```

Figure 1:Code output Example

```
out of whileloop : added
out of whileloop : added
out of whileloop : added
 out of whileloop : added
out of whileloop : added
out of whileloop : added
out of whileloop : added
 out of whileloop : added
out of whileloop : added
out of whileloop : added
```

Command: time python3 pump-fixed.py 10

```
Kens-MBP:concurrency_beyond_locks ken$ time python3 pump-fixed.py 10
starting Consumer
starting Producer
"Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was rig
ht but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly h
ere to help with that part of your education; after all, when you go to interview at Google, and the
y ask you a trick question about how to use semaphores, it might be good to actually know what a sem
aphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is
to get you interested in something, to learn something more about the subject matter on your own and
 not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s
dad, Vedat Arpaci) used to say, “Learn beyond the classroom”.
real    0m1.476s
user    0m1.449s
sys     0m0.024s
```

```
out of whileloop : added
out of whileloop : added
```

Figure 2: A piece of output running pump-fixed.py with four additional print statement included in the deadlock.py